SCELBAL UPDATE

ISSUE 05 - 6/77

UNLIMITED! (WELL - ALMOST) VARIABLE NAMES!

One of the improvements most often suggested for SCELBAL is to increase the number of variable names allowed. The original version allowed a total of 20 regular variable names. It was possible to increase the effective number of variables in a system having DIM capability installed, but even when performing "tricks" such as that, the number of variable names was limited to a maximum of 84. A good many users felt it would be nice to substantually increase the number of variable names allowed in a program - without having to snitch from elements in an array.

O.K.! Here it is - a modification to SCELBAL that will theoretically allow you to have as many variables as can be defined by valid two character symbolic names, provided you have enough memory in your system to store all the variables desired!

Essentially, the modification changes SCELBAL so that it stores variable names and their values starting at the top (highest allowable address value) of the User's Program Buffer and works downward toward the source code in the buffer which is stored in ascending address values as new lines are entered. The variable names table previously assigned to Page 27 starting at Location 210 is no longer used if the user elects to install this modification.

Listings of the modification for both 8008 and 8080 machines are included. The routines shown may be simply "overlaid" over the original routines.

Several notes of caution are in order. First, the modification as shown in the accompanying listings is for the essentially unmodified version of SCELBAL as presented in the basic publication. If you have made modifications to your version - be careful. Same goes if you have implemented any of the supplements.

In particular, if you have been playing around with compacting SCELBAL for an 8080 machine and have changed the order of the bytes stored in the End of User Program Buffer Pointer (Page 26, Locations 364, 365) as mentioned in SCELBAL UPDATE Issue 04, you will have to change things around a little bit in the accompanying listing in the vicinity of the LOOKU3 subroutine at Page 05 Location 157 etc.

If you have installed Strings or Mathematical Supplements, or if your User Program Buffer storage area does not end at Page 54 Location 377 in your system, you will need to alter the values in the accompanying listing marked with a "$$" notation in the comments section (such as Page 05 Location 54 and Page 11 Location 44) so that the end of the User Program Buffer storage area is set up properly by the new unlimited variables modification routines.

It is assumed that those who have otherwise modified SCELBAL or relocated the program, will know how to proceed to adapt the modification.

Finally, a note of caution. The modification checks to see that variables do not run into a user's source listing. However, no check is made to see that the user buffer does not run into the variables table. It is thus theoretically possible to "bomb" the variables table if one was, for instance, inserting new lines into a source listing and alternating with the RUN mode to test the operation of the program being developed. If it looks like storage will be tight in a program; load the source *entirely* before executing a RUN command! Since variable names are added to the variables table as a program is executed, the modified program will indicate if buffer space is exhausted.

Have fun with the new capability!

LISTING FOR AN 8008

```
000 000                 /
000 000                 ORG 005 033
005 033                 /
005 033  106 045 005    LOOK, CAL NEWVT     /CALL NEW VAR STORAGE RTN
005 036  240            NDA                 /CHECK STATUS ON RETURN
005 037  150 155 010    JTZ LOOKU4          /IF FOUND MATCH IN TBL - PROC
005 042  104 135 010    JMP LOOK3A          /IF HAVE EOT - ADD ENTRY TO V
005 045                 /
005 045  066 120        NEWVT, LLI 120      /POINTER TO SYMBOL
005 047  056 026        LHI 026             /**BUFFER STORAGE AREA
005 051  046 377        LEI 377             /POINTER TO START OF
005 053  036 054        LDI 054             /$$ NEW VARS STORAGE AREA
005 055  307            LAM                 /FETCH (CC) OF STRING IN BFR
005 056  074 001        CPI 001             /SEE IF IT IS EQUAL TO ONE
005 060  110 067 005    JFZ LOOKUA          /JUMP AHEAD IF NOT EQUAL TO O
005 063  066 122        LLI 122             /ELSE SET PNTR AND CLEAR 2ND
005 065  076 000        LMI 000             /BYTE OF NAME TO ZERO
005 067  353            LOOKUA, LHE         /SET POINTER TO
005 070  364            LLE                 /FIRST LOCATION
005 071  307            LAM                 /IN VARIABLES TABLE
005 072  240            NDA                 /SEE IF EQUAL TO ZERO
005 073  150 150 005    JTZ LOOKU3          /IF SO, NOTHING IN TABLE
005 076                 /
005 076  066 121        LOOKU1, LLI 121     /SET POINTER TO 1ST CHARACTER
005 100  056 026        LHI 026             /**OF NAME IN THE SYMBOL BFR
005 102  106 356 022    CAL SWITCH          /SAVE IN D&E AND FETCH
005 105  307            LAM                 /POINTER TO VT, THEN FETCH
005 106  061            DCL                 /FIRST ENTRY TO THE ACC
005 107  317            LBM                 /AND 2ND ENTRY TO REG B
005 110  106 164 003    CAL DEC             /DECREMENT VT PNTR ONCE MORE
005 113  106 356 022    CAL SWITCH          /SAVE VT POINTEP AND GET SB
005 116  277            CPM                 /POINTER. SEE IF HAVE SAME
005 117  110 132 005    JFZ LOOKU2          /NAME. TO NEXT ENTRY IF
005 122  060            INL                 /NOT. BUT, IF FIRST LETTER
005 123  301            LAB                 /MATCHES - THEN TRY
005 124  277            CPM                 /SECOND. IF FIND NAME
005 125  110 132 005    JFZ LOOKU2          /MATCHES CAN STORE VALUE
005 130  250            XRA                 /SO CLEAR ACC TO INDICATE
005 131  007            RET                 /MATCH, THEN RETURN TO CALLER
005 132                 /
005 132  016 004        LOOKU2, LBI 004     /PUT 4 INTO REGISTER B
005 134  353            LHD                 /FETCH VARIABLES TABLE
005 135  364            LLE                 /POINTER INTO REGS H&L
005 136  106 113 003    CAL SUBHL           /SUBTRACT 4 FROM PNTR VALUE
005 141  307            LAM                 /FETCH FM ADDR POINTED TO
005 142  335            LDH                 /SAVE VARIABLES TABLE
005 143  346            LEL                 /POINTER IN D&E
005 144  240            NDA                 /TEST LAST BYTE FROM VT
005 145  110 076 005    JFZ LOOKU1          /IF NOT EOT, CONT SEARCH
005 150                 /
005 150  016 006        LOOKU3, LBI 006     /IF FOUND EOT
005 152  106 113 003    CAL SUBHL           /SUBTRACT 6 FROM PNTR AND
```

```
005 155  335            LDH              /SAVE VARIABLES TABLE
005 156  346            LEL              /POINTER IN D&E
005 157  056 026        LHI 026          /**SET POINTER TO END    ****
005 161  066 364        LLI 364          /OF USER PROGRAM BUFFER ****
005 163  307            LAM              /FETCH EOB PAGE VALUE
005 164  273            CPD              /COMPARE WITH VT PNTR VALUE
005 165  160 176 005    JTS OKDOK2       /IF POS HERE, NO CONFLICT
005 170  060            INL              /IF NOT, FETCH LOW ADDR
005 171  307            LAM              /OF END OF USER PGM BF PNTR
005 172  274            CPE              /AND TEST FOR ROOM ON PAGE
005 173  100 222 002    JFC BIGERR       /IF NOT, HAVE AN ERROR!
005 176  106 356 022    OKDOK2, CAL SWITCH/IF OK, RESTORE VT PNTR
005 201  076 000        LMI 000          /TO H&L AND MAKE EOT MARKER
005 203  106 174 003    CAL INDEXB       /ADD 6 BACK TO VT PNTR
005 206  106 356 022    CAL SWITCH       /SAVE VT PNTR IN D&E
005 211  066 121        LLI 121          /SET PNTR TO 1ST CHAR IN SB
005 213  307            LAM              /FETCH 1ST CHARACTER TO ACC
005 214  060            INL              /ADVANCE BUFFER POINTER
005 215  317            LBM              /FETCH 2ND CHAR TO REG B
005 216  353            LHD              /GET VARIABLES TABLE
005 217  364            LLE              /POINTER IN H&L
005 220  370            LMA              /STORE SYMBOL NAME
005 221  061            DCL              /IN THE VARIABLES TABLE
005 222  371            LMB              /- BOTH CHARACTERS -
005 223  006 377        LAI 377          /SET ACC TO ALL ONES TO FLAG
005 225  007            RET              /JOB DONE, RETURN TO CALLER
005 226                 /
000 000                 ORG 010 100
010 100                 /
010 100  106 045 005    STOSY1, CAL NEWVT /CALL NEW VAR STORAGE RTN
010 103  240            NDA              /CHECK STATUS ON RETURN
010 104  150 117 010    JTZ STOSY4       /IF FOUND MATCH - PROCESS
010 107  016 004        LBI 004          /IF HAVE EOT THEN SET UP
010 111  106 113 003    CAL SUBHL        /TO ADD ENTRY
010 114  104 127 010    JMP STOSY5       /TO THE VARIABLES TABLE
010 117                 /
010 117  106 356 022    STOSY4, CAL SWITCH/RESTORE VT POINTER TO H&L
010 122  016 003        LBI 003          /LOAD 3 INTO REG B
010 124  106 113 003    CAL SUBHL        /SUBTRACT 3 FROM VT PNTR
010 127                 /
010 127  106 255 022    STOSY5, CAL FSTORE/FPACC INTO VT LOCATIONS
010 132  104 255 002    JMP CLESYM       /CLEAR SYMBOL BF & EXIT
010 135                 /
010 135  250            LOOK3A, XRA      /CLEAR THE ACCUMULATOR
010 136  106 164 003    CAL DEC          /AND PLACE
010 141  370            LMA              /ZERO
010 142  061            DCL              /INTO
010 143  370            LMA              /THE
010 144  106 164 003    CAL DEC          /VAPIABLES
010 147  370            LMA              /TABLE
010 150  061            DCL              /FOR THE
010 151  370            LMA              /INITIAL VALUE
010 152  104 165 010    JMP LOOKU5       /GO FINISH UP
010 155                 /
010 155  106 356 022    LOOKU4, CAL SWITCH/POINTER TO VT INTO H&L
010 160  016 003        LBI 003          /COUNT OF 3 INTO REG B
010 162  106 113 003    CAL SUBHL        /SUBTRACT 3 FROM VT PNTR
010 165                 /
010 165  106 317 022    LOOKU5, CAL SAVEHL/SAVE VT POINTER
010 170  066 227        LLI 227          /SET UP PNTR TO ARITHMETIC
010 172  056 001        LHI 001          /**STACK POINTER
010 174  307            LAM              /FETCH POINTER VALUE
010 175  004 004        ADI 004          /ADD 4 FOP NEW ENTRY
010 177  370            LMA              /RESTORE STACK POINTER
010 200  360            LLA              /AND SET UP NEW AS VALUE
010 201  106 255 022    CAL FSTORE       /PUT THE FPACC ON THE AS
010 204  106 337 022    CAL RESTHL       /RESTORE VT POINTER
010 207  106 244 022    CAL FLOAD        /PUT THE VAR INTO FPACC
010 212  104 231 005    JMP PARSE        /TO THE PARSE ROUTINE
010 215                 /
010 215                 ORG 011 041
011 041                 /
011 041  066 377        LLI 377          /POINTEB TO START OF
011 043  056 054        LHI 054          /$$ NEW VAR'S STORAGE AREA
011 045  300            LAA              /REPLACE WITH NOP INSTRUC
011 046                 /
```

LISTING FOR AN 8080

```
003 165                 ORG 005 033
005 033                 /
005 033  315 045 005    LOOK, CAL NEWVT   /CALL NEW VAR STORAGE RTN
005 036  247            NDA               /CHECK STATUS ON RETURN
005 037  312 155 010    JTZ LOOKU4        /IF FOUND MATCH IN TBL - PROCESS
005 042  303 135 010    JMP LOOK3A        /IF HAVE EOT - ADD ENTRY TO VT
005 045                 /
005 045  056 120        NEWVT, LLI 120    /POINTER TO SYMBOL
005 047  046 026        LHI 026           /**BUFFER STORAGE AREA
005 051  036 377        LEI 377           /POINTEP TO START OF
005 053  026 054        LDI 054           /$$ NEW VARS STORAGE AREA
005 055  176            LAM               /FETCH (CC) OF STRING IN BFR
005 056  376 001        CPI 001           /SEE IF IT IS EQUAL TO ONE
005 060  302 067 005    JFZ LOOKUA        /JUMP AHEAD IF NOT EQUAL TO ONE
005 063  056 122        LLI 122           /ELSE SET PNTR AND CLEAR 2ND
005 065  066 000        LMI 000           /BYTE OF NAME TO ZERO
005 067  142            LOOKUA, LHD       /SET POINTER TO
005 070  153            LLE               /FIRST LOCATION
005 071  176            LAM               /IN VARIABLES TABLE
005 072  247            NDA               /SEE IF EQUAL TO ZERO
005 073  312 150 005    JTZ LOOKU3        /IF SO, NOTHING IN TABLE
005 076                 /
005 076  056 121        LOOKU1, LLI 121   /SET POINTER TO 1ST CHARACTER
005 100  046 026        LHI 026           /**OF NAME IN THE SYMBOL BFR
005 102  315 356 022    CAL SWITCH        /SAVE IN D&E AND FETCH
005 105  176            LAM               /POINTEP TO VT, THEN FE
005 106  055            DCL               /FIPST ENTPY TO THE ACC
005 107  106            LBM               /AND 2ND ENTRY TO REG B
005 110  315 164 003    CAL DEC           /DECREMENT VT PNTR ONCE
005 113  315 356 022    CAL SWITCH        /SAVE VT POINTER AND GE
005 116  276            CPM               /POINTER. SEE IF HAVE S
005 117  302 132 005    JFZ LOOKU2        /NAME. TO NEXT ENTRY IF
005 122  054            INL               /NOT. BUT, IF FIRST LET
005 123  170            LAB               /MATCHES - THEN TRY
005 124  276            CPM               /SECOND. IF FIND NAME
005 125  302 132 005    JFZ LOOKU2        /MATCHES CAN STORE VALU
005 130  257            XRA               /SO CLEAR ACC TO INDICA
005 131  311            RET               /MATCH, THEN RETURN TO
005 132                 /
005 132  006 004        LOOKU2, LBI 004   /PUT 4 INTO REGISTER B
005 134  142            LHD               /FETCH VARIABLES TABLE
005 135  153            LLE               /POINTER INTO REGS H&L
005 136  315 113 003    CAL SUBHL         /SUBTRACT 4 FROM PNTR V
005 141  176            LAM               /FETCH FM ADDR POINTED
005 142  124            LDH               /SAVE VARIABLES TABLE
005 143  135            LEL               /POINTER IN D&E
005 144  247            NDA               /TEST LAST BYTE FROM VT
005 145  302 076 005    JFZ LOOKU1        /IF NOT EOT, CONT SEARC
005 150                 /
005 150  006 006        LOOKU3, LBI 006   /IF FOUND EOT
005 152  315 113 003    CAL SUBHL         /SUBTRACT 6 FROM PNTR A
005 155  124            LDH               /SAVE VARIABLES TABLE
005 156  135            LEL               /POINTER IN D&E
005 157  046 026        LHI 026           /**SET POINTER TO END
005 161  056 364        LLI 364           /OF USER PROGRAM BUFFER
005 163  176            LAM               /FETCH EOB PAGE VALUE
005 164  272            CPD               /COMPARE WITH VT PNTR V
005 165  372 176 005    JTS OKDOK2        /IF POS HERE, NO CONFLI
005 170  054            INL               /IF NOT, FETCH LOW ADDR
005 171  176            LAM               /OF END OF USER PGM BF
005 172  273            CPE               /AND TEST FOR ROOM ON P
005 173  322 222 002    JFC BIGERR        /IF NOT, HAVE AN ERROR!
005 176  315 356 022    OKDOK2, CAL SWITCH/IF OK, RESTORE VT PNTR
005 201  066 000        LMI 000           /TO H&L AND MAKE EOT MA
005 203  315 174 003    CAL INDEXB        /ADD 6 BACK TO VT PNTR
005 206  315 356 022    CAL SWITCH        /SAVE VT PNTR IN D&E
005 211  056 121        LLI 121           /SET PNTR TO 1ST CHAR I
005 213  176            LAM               /FETCH 1ST CHARACTER TO
005 214  054            INL               /ADVANCE BUFFER POINTER
005 215  106            LBM               /FETCH 2ND CHAR TO REG
005 216  142            LHD               /GET VARIABLES TABLE
005 217  153            LLE               /POINTER IN H&L
005 220  167            LMA               /STORE SYMBOL NAME
005 221  055            DCL               /IN THE VARIABLES TABLE
005 222  160            LMB               /- BOTH CHARACTERS -
005 223  076 377        LAI 377           /SET ACC TO ALL ONES TO
005 225  311            RET               /JOB DONE, RETURN TO CA

002 223                 ORG 010 100
010 100                 /
010 100  315 045 005    STOSY1, CAL NEWVT /CALL NEW VAP STORAGE R
010 103  247            NDA               /CHECK STATUS ON RETURN
010 104  312 117 010    JTZ STOSY4        /IF FOUND MATCH - PROCE
010 107  006 004        LBI 004           /IF HAVE EOT THEN SET U
010 111  315 113 003    CAL SUBHL         /TO ADD ENTRY
010 114  303 127 010    JMP STOSY5        /TO THE VAPIABLES TABLE
010 117                 /
010 117  315 356 022    STOSY4, CAL SWITCH/RESTORE VT POINTER TO
010 122  006 003        LBI 003           /LOAD 3 INTO REG B
010 124  315 113 003    CAL SUBHL         /SUBTRACT 3 FROM VT PNT
010 127                 /
010 127  315 255 022    STOSY5, CAL FSTORE/FPACC INTO VT LOCATION
010 132  303 255 002    JMP CLESYM        /CLEAP SYMBOL BF & EXIT
010 135                 /
010 135  257            LOOK3A, XRA       /CLEAP THE ACCUMULATOR
010 136  315 164 003    CAL DEC           /AND PLACE
010 141  167            LMA               /ZERO
010 142  055            DCL               /INTO
010 143  167            LMA               /THE
010 144  315 164 003    CAL DEC           /VARIABLES
010 147  167            LMA               /TABLE
010 150  055            DCL               /FOR THE
010 151  167            LMA               /INITIAL VALUE
010 152  303 165 010    JMP LOOKU5        /GO FINISH UP
010 155                 /
010 155  315 356 022    LOOKU4, CAL SWITCH/POINTER TO VT INTO H&L
010 160  006 003        LBI 003           /COUNT OF 3 INTO REG B
010 162  315 113 003    CAL SUBHL         /SUBTRACT 3 FROM VT PNT
010 165                 /
010 165  315 317 022    LOOKU5, CAL SAVEHL/SAVE VT POINTER
010 170  056 227        LLI 227           /SET UP PNTR TO ARITHME
010 172  046 001        LHI 001           /**STACK POINTER
010 174  176            LAM               /FETCH POINTER VALUE
010 175  306 004        ADI 004           /ADD 4 FOR NEW ENTRY
010 177  167            LMA               /RESTORE STACK POINTER
010 200  157            LLA               /AND SET UP NEW AS VALU
010 201  315 255 022    CAL FSTORE        /PUT THE FPACC ON THE A
010 204  315 337 022    CAL RESTHL        /RESTORE VT POINTER
010 207  315 244 022    CAL FLOAD         /PUT THE VAR INTO FPACC
010 212  303 231 005    JMP PARSE         /TO THE PARSE ROUTINE
010 215                 /
010 215                 ORG 011 041
011 041                 /
011 041  056 377        LLI 377           /POINTEB TO START OF
011 043  046 054        LHI 054           /$$ NEW VAR'S STORAGE A
011 045  177            LAA               /REPLACE WITH NOP INSTR
011 046                 /
```

EXTENDED MATHEMATICAL FUNCTIONS AVAILABLE

Five extended mathematical functions are now available for SCELBAL. The new functions, made available as a supplemental publication, provide users with the following capabilities when installed: SIN, COS, EXP(e), LOG(e), and ATN.

The SIN and LOG functions are calculated using Chebyshev optimized Taylor series. The EXP and ATN are calculated using continued fractions. The COS function is calculated using the SIN function. The argument of any function is reduced to an interval where the Taylor series or continued fractions is reasonably accurate. The argument range for the functions are as follows:

SIN -4194303<X<4194303
COS -4194303<X<4194303
EXP -89<X<89
LOG X>0
ATN -1E37<X<1E37

The supplemental booklet contains source and object listings as in other publications related to SCELBAL. The assembled object listings provided reside in locations on pages 50 through 54. They may be reassembled to reside elsewhere by the user if desired. String Function users should note that those same pages are used by sections of the String Functions.

The price of the Mathematical Supplement to SCELBAL is $5.00 in the U.S. including U.S. mail delivery. Foreign purchasers should include $2.00 for airmail delivery of the supplement.

A FEW CORRECTIONS

C. A. Bannister of Richmond, VA, was the first to report some object code errors in the listing for modified SCELBAL shown on page 3 of SCELBAL UPDATE Issue 02. The object code errors only occurred in the 8008 listing.

It seems that the object codes for XRA, LMA and LLA directives got fouled up in the listing. The code for XRA should be 250, for LMA it is 370 and for LLA it is 360.

Alert Bannister also noted a typographical error on the first line of Mr. Toy's routine shown on page 2 of Issue 04: The code. for LLI should be 066 not 006 as printed.

Thanks for the use of your sharp eyes - and our apologies to our readers for letting those errors get by. — Ed.

STRINGS PATCH

Mr. H. J. Lewis of Canada has spotted a glitch in the Strings Supplement. The following patch, (named in his honor!) should be installed at Page 50 Location 327:

JFZ HJLFIX

It will replace the JFZ SSTRCL instruction. The patch, which may be placed on Page 54 at Location 301, is just two instructions:

```
HJLFIX,  CAL SWITCH
         JMP SSTRCL
```

This patch will correct an anomaly in the string comparison routines that can effect string comparison operations.

Many thanks to Mr. Lewis for his persistence in analyzing and solving this problem and bringing it to our attention! —Ed.

MATHEMATICAL FUNCTIONS THE OTHER WAY!

One of your fellow SCELBAL users, Robert Leonard, 3003 Driscoll Drive, San Diego, CA. 92117, sent in a nice set of subroutines to calculate the sine, cosine, tangent, arc tangent, log and exponent. The LOG and EXP functions he provided are natural base. The trig functions expect the angles to be given in radians. The variable names assigned and line numbers of the various routines he provides are summarized as follows:

SIN(X) = SN	GOSUB 10
COS(X) = CS	GOSUB 20
TAN(X) = TN	GOSUB 30
ATN(X) = AT	GOSUB 40
LOG(X) = LG	GOSUB 80
EXP(X) = EX	GOSUB 100

The subroutines making up the high level package are shown alongside this column.

Robert also mentioned that he likes to use a patch to eliminate the decimal point and zero after whole numbers. Says he likes the format for its neatness in games, etc. If you want to take a look at it, the patch he uses is presented here:

```
025 147      JMP PATCH

PATCH,       LLI 166
             LAM
             NDI 370
             RTZ
             LAI 256
             CAL ECHO
             JMP NODECP
```

Thanks for the very nice high level math package Robert. Hope you enjoy the check we have sent you for your efforts! - Ed.

LISTING OF HIGH LEVEL MATHEMATICAL FUNCTIONS

```
10 Z=X
11 SN=X
12 N=2
13 Z=-Z*(X↑2)/(N*(N+1))
14 SN=SN+Z
15 N=N+2
16 IF ABS(Z)>.0001 THEN 13
17 RETURN
20 Z=1
21 CS=1
22 N=1
23 Z=-Z*(X↑2)/(N*(N+1))
24 CS=CS+Z
25 N=N+2
26 IF ABS(Z)>.0001 THEN 23
27 RETURN
30 GOSUB 10
31 GOSUB 20
32 TN=SN/CS
33 RETURN
40 IF X<.7 THEN 60
41 IF X>1.4 THEN 70
42 Y=X/SQR(1+(X↑2))
43 Z=Y
44 AT=Y
45 N=1
46 Z=Z*(Y↑2)*(N↑2)/((N+1)*(N+2))
47 AT=AT+Z
48 N=N+2
49 IF ABS(Z)>.000001 THEN 46
50 RETURN
60 Z=X
61 AT=X
62 N=3
63 Z=SGN(Z)*(-(X↑N)/N)
64 AT=AT+Z
65 N=N+2
66 IF ABS(Z)>.000001 THEN 63
67 RETURN
70 Z=1.570796
71 AT=Z
72 N=1
73 Z=SGN(Z)*(-1/(N*(X↑N)))
74 AT=AT+Z
75 N=N+2
76 IF ABS(Z)>.000001 THEN 73
77 RETURN
80 Y=0
81 IF X<1 THEN 85
82 X=X/2
83 Y=Y+1
84 GOTO 81
85 IF X>.5 THEN 89
86 X=2*X
87 Y=Y-1
88 GOTO 85
89 X=(X-.707107)/(X+.707107)
90 LG=2*((X)+((X↑3)/3)+((X↑5)/5)+((X↑7)/
7))-.346573
91 LG=LG+(Y*.693147)
92 RETURN
100 Z=1
101 EX=1
102 N=1
103 Z=Z*X/N
104 EX=EX+Z
105 N=N+1
106 IF ABS(Z)>.000001 THEN 103
107 RETURN
```

What is the VALUE of VAL?

String functions are designed to allow the user to manipulate "strings" of alphanumeric characters *instead of mathematical quantities.*

However, there may be times when it is desirable to manipulate information in essentially two forms - as a string of characters, and as a *numerical value.*

Suppose, for instance, one wanted to have the computer make a list of groceries showing the price for each item, and then also *mathematically* sum the prices to obtain a total?

TOMATOES	24
LETTUCE	79
CARROTS	38
ORANGES	98

One could use string capabilities to list the items and their prices. But the character strings themselves are useless for calculating mathematical information *unless* one has the special capability to convert between one mode and the other. *That is what the VAL function in the SCELBAL String Supplement provides!*

The VAL function *converts characters in a string from an ASCII representation of a decimal number to its numeric value. In other words, the prices in the example can be converted from character string format to actual numeric values that can be mathematically manipulated by SCELBAL!*

Assume the lines in the above example are each composed of two strings 'A$' (item) and 'B$' (price). The 'price' strings in the example would be elements in string arrays B$(1) through B$(4). One could obtain a numerical value for the total of all the prices in the list with a routine such as:

```
FOR X = 1 TO 4
LET T = VAL(B$(X)) + T
NEXT X
PRINT T
```

This is because the VAL function would convert the numerical character strings *to mathematical VALUES!*

If reader interest warrants, we will discuss capabilities of the String Supplement for SCELBAL some more in the next issue of this publication

SCELBI COMPUTER CONSULTING, INC.
1322 Rear — Boston Post Road
Milford, CT 06460

U. S. POSTAGE
FIRST CLASS
PAID
PERMIT No. 481
MILFORD, CONN.

129

F. R. RUCKDESCHEL
(XEROX CORP)
800 PHILLIPS ROAD
WEBSTER, NY 14580

FIRST CLASS MAIL!

SHIP TO

AME: _____

DDR: _____

TY: _____

ZIP

Please note: All foreign orders must be accompanied by bank or postal money order payable in U.S. funds.

INDIVIDUALS SHOULD SUBMIT PAYMENT IN FULL BY CHECK OR MONEY ORDER — OR USE YOUR "MASTERCHARGE" ACCT.
"MASTERCHARGE"
ACCOUNT NUMBER .....

BANK # ..... EXPIRATION DATE .....

SIGNATURE .....

QTY	DESCRIPTION	UNIT PRICE	TOTAL PRICE

MERCHANDISE TOTAL .....

CONNECTICUT RESIDENTS ADD 7% STATE TAX .....

SHIPPING CHARGES .....

AMOUNT ENCLOSED .....

ORDER FORM

SCELBAL UPDATE

ISSUE 06 - 3/78

SCELBAL—II READY FOR RELEASE

For sometime there has been a question as to whether or not SCELBAL-II would ever be released in source format. In appreciation of our early customers, a compromise has been reached. As detailed in a separate flyer that will accompany this edition of SCELBAL UPDATE, the revised edition developed specifically for 8080/Z-80 systems will be made available to registered SCELBAL owners for a modest fee as an *uncommented* assembled source listing. Since SCELBAL-II essentially follows the general structure of the original version, SCELBAL owners with 8080 or Z-80 systems should find the improved version attractive and understandable. Those not having the original SCELBAL documentation would likely find it somewhat discouraging to attempt to decipher the uncommented listing of SCELBAL-II. In any event, SCELBAL-II will only be made available to purchasers of the original SCELBAL documentation.

THIS TO BE LAST ISSUE OF SCELBAL UPDATE

As we indicated when we began publication of this journal, the objectives of this supplementary publication were multiple-purpose. First, it would provide a vehicle for informing SCELBAL customers of program corrections that were liable to be required in a program the size and scope of an interpreter. Second, it would be an experimental publication to determine if users wanted to work through the publication to amplify the package in any way. We said we would provide this publication, free for a limited period of time, and possibly on a subscription basis thereafter, if users showed this is what they wanted.

Well, the free period is over, and support for such a publication on a subscription basis has not been demonstrated. Only a handful of readers have submitted material for publication even though an honorarium is presented for published material. Only a fraction of a percent or readers have expressed any interest in having this publication continue on a subscription basis.

The journal has lived up to its task of informing SCELBAL users of program bugs discovered by users over a more than sufficient time span. SCELBAL, with minor alterations pointed out in this journal, is a proven interpretive language.

Best wishes to all its users!

BOWLING HANDICAPPER IN ONLY 512 BYTES!

Harold F. Bower has been running SCELBAL in an eight K 8008 system for some time so he had a limited 512 bytes of user program storage room. That didn't stop him though. He sent in the following program that has been helping him calculate information used by bowling leagues

```
10  INPUT A                          Input total games to date
20  PRINT "INPUT SCORES";
30  INPUT B,C,D                      Input scratch scores
40  PRINT "SCR TOT";
50  INPUT F                          Input previous scratch total
60  PRINT "HDCP TOT";
70  INPUT G
80  PRINT "TOT";
90  INPUT H                          Input previous total pins -
                                     keeping this list eases problems
                                     with changing players in singles
                                     leagues
100 PRINT "HDCP";                    Input player's previous
110 INPUT I                          handicap
115 PRINT
120 PRINT B+C+D;TAB(12);3*I;TAB(24);3*I+B+C+D
130 PRINT "-----";TAB(12);"-----";TAB(24);"-----"
140 PRINT F+B+C+D;TAB(12);G+3*I;TAB(24);H+B+C+D+3*I
                                     The above three lines give
                                     formatted output of scratch
                                     total, handicap total, and
                                     cumulative total suitable for
                                     a 32 column TV display
150 PRINT (F+B+C+D)/A;TAB(12);.66667*(190 - (F+B+C+D)/A)
                                     The above line prints the new
                                     average and handicap
160 GOTO 20                          If next player has bowled the
                                     same number of games change
                                     this to go to line 10
170 END
```

Harold says that while the above program requires quite a few more manual entries than would be required if master files were maintained in string variable format, and could be saved then later loaded and modified with the new results being saved for the next time, the program does save a considerable amount of work and can be run on a minimal system.

Howard is stationed in Germany at HQ 5th SIG CMD, DCSOPS—TD, APO New York, NY 09056. He has recently upgraded his system to a 12K Z-80 so he should really be cranking out handicaps by this time!

MORE FOR BAUDOT MACHINE USERS

Mr. S. J. Toy, a frequent contributor to this publication, still runs a SCELBI 8008 system with a baudot teletype machine for basic I/O. He recently sent in some more information on his modifications of SCELBAL to facilitate its use with a baudot device.

"A while back I described some modifications I made to the INPUT portion of SCELBAL. [See Update Issue 02 — Ed.] Since that time I have discovered that it would not work with the CHR function, mainly because the latter follows a different route through SCELBAL. To overcome this I have made several changes that now make INPUT even more useful.

To allow more than one item of data to be input on the same line, the CR key obviously cannot be used to terminate the entry. Instead, I use another key, which in my case is the Blank key on my model 15 TTY. The STRINF routine is rearranged so that CRLF is skipped when the blank key is used. My previous changes on page 017 that substitute a semicolon for the comma have been removed, and all routines there are restored to their original form. While this allows more than one input per line on the TTY, it also requires that the end of the line be terminated by a following PRINT statement. This seems to be a good tradeoff. The CR key can be used at the end of the line but it is probably better to use a PRINT statement, which makes the carriage return automatic. My modifications to INPUT now consist only of the following:

```
003 046   ***                          Code for Blank key which replaces code for Control/C.

003 050   105 003                      Address in re-arranged STRINF routine to skip CRLF op.

003 102   106 141 003   STRINF,  CAL CRLF
003 105   312                      LBC
003 106   106 113 003            CAL SUBHL
003 111   372                      LMC
```

If one wishes to retain Control/C the test for Line Feed can be sacrificed instead, since LF is not normally used during input of data.

To input data into the same line as data being printed out from memory under TAB control, it is necessary to increment the COLUMN COUNTER each time a digit is input. This is accomplished by inserting a column counter incrementing routine into CINPUT, which is provided by the user for his own particular input device.....By adding a test for the Blank key and the Delete key, which are both non-printing, the column counter incrementing routine can be skipped. If this is not done, the position of the column will be displaced by one character, although this can be compensated for by changing the TAB value. Skipping the column counter incrementer, however, is better, as it simplifies programming. The complete routine to be inserted into CINPUT.....that I use.....is as follows:

```
074 ——        CPI  ——          Test for Blank key.
150 —— ——     JTZ  —— ——       Skip col cnt increment if Blank.
074 ——        CPI  ——          Test for Delete key.
150 —— ——     JTZ  —— ——       Skip col cnt increment if Delete.
066 043       LLI  043         Point to Column Counter.
056 001       LHI  001         "  "  "  "
317           LBM              Load column cntr into B.
010           INB              Increment column counter.
371           LMB              Restore column cntr to memory.
```

The code for the Blank key or the Delete key is in the accumulator when the routine is entered. If either JTZ is true, the jump is to the byte immediately following the end of the routine, which effectively bypasses the column counter incrementer. Incidently, the Delete key, in my case is the BELL key of the model 15 TTY.....

One needs to be careful that registers B, H, and L are free when the routine is used. Locating the routine here covers both numerical and CHR inputs. This addition is useful only if the preceeding modification to INPUT is made.

Another improvement I have made to SCELBAL is to add a function to limit the number of digits printed out. This has been a problem in printing tables of data where either allowance must be made for printing out the full 7 digits or accept an occasional overlap between columns. The INTEGER function does not seem to work for numbers with more than 4 digits [a result of binary rounding operations that start to show their affect when numbers exceed 4 digits — Ed.], and in any case can be used only with whole numbers. Even a number-rounding routine does not always work because the last stage of division frequently results in the value extending back out to 7 digits.

My new function changes the value at location 025 035 which specifies the number of digits to be printed. It replaces the SGN function, which I have never used, and occupies the same space with one byte left over. The Function Names Table is also changed to DIG. The subscript of DIG is the number of digits to be printed. A user program statement would take the form of:

100 PRINT DIG(3)

This will limit all values to three significant digits, until a subsequent statement changes the limit. Besides the 3 digits, allowance must be made, of course, for a possible minus sign and a decimal point. A listing for the Digits Function follows:

```
007 360   106 000 020   DIGX,  CAL FPFIX     Cvrt FP to fixed.
007 363   066 124               LLI 124      Point to LSW.
007 365   307                   LAM          Load to Acc.
007 366   066 035               LLI 035      Point to digits
007 370   056 025               LHI 025      Number storage.
007 372   370                   LMA          Load new nmbr.
007 373   104 010 010           JMP 010 010  Jump to suppress
                                             printout of nmbr
                                             and to return.

026 305   304                   304          ASCII "D"
026 306   311                   311          ASCII "I"
026 307   307                   307          ASCII "G"
```

[Thanks for all the new information. We have had a number of people ask about a modification to restrict the number of digits outputted. Your's looks like a real straightforward technique to use! — Ed.]

TEXT CONTROL & TRACE CAPABILITY SUBMITTED BY SCELBAL USER

Robert Pearce of 504 McCoys Fork Rd, Walton, KY 41094, says he is not a technical writer but he took the time to send in some pretty clear explanations of how he added some "extra" capabilities to SCELBAL. We think his additions will be of interest to many SCELBAL users.

The first improvement he discusses is a modification to the TEXTC routine that he has named TEXTCM. The modification provides the user with the capability of halting a listing of a program at any time by depressing any character on the input keyboard (except C/R or CTRL/C). Doing so places the program in an "input loop" effectively halting operations while the user inspects the system's display. To continue the display the user may type

a C/R (carriage return). Or, to end the listing and return to the EXECutive routine the user can enter CTRL/C.

Naturally, this capability will be super for those using a CRT display who need capability for displaying a section of the user program buffer at a time. And, it is valuable for any user in that it allows the termination of a long listing when a point of interest has been reached.

The second improvement he presents provides program trace capability. It requires the insertion of a patch at the routine labeled SYNTX3. When trace is activated SCELBAL will display the line number of each line executed in a user's program. Trace capability is controlled using a switch activated via a UDF function.

Robert notes that coupling the trace capability with the TEXTCM modification provides a powerful debugging combination.

He also mentions that his version of SCELBAL has been implemented in a MIKE-2 system.

A commented source listing of the modifications required to implement his improvements is shown below.

```
TEXTCM,  LCM          Fetch (cc) from the first location in
         LAM          The buffer (H&L pnting there)
         NDA          Into Reg C & A. Test the (cc) value.
         RTZ          No display if (cc) is zero.
TEXTCL,  CAL ADV      Advance pointer to next location.
         LAM          Get character from buffer.
         CAL ECHO     Display character.
         IN *         Get input from keyboard.
         CPI 000      Test for 0.
         JTZ END      If yes, continue with TEXTC rtn.
INLOOP,  CAL INPUT    (User subrtn without echo) stop here.
         CPI c/r      And wait for a C/R or a CTL/C.
         JTZ END      If get C/R, continue with display.
         CPI ctl/c    If get CTL/C exit to
         JTZ EXEC     Start over.
         JMP INLOOP   Else cycle.
END,     DCC          Decrement (cc).
         JFZ TEXTCL   If (cc) is not zero continue display.
         RET          Exit to calling routine.

        [AT PAGE 02 LOCATION 061 CHANGE:]
SYNTX3,  CAL TRACE    Insert TRACE patch call.

        [AT A SUITABLE PATCH AREA ADD:]
TRACE,   LLI 201      Replace SYNTX3 instructions.
         LBM
SWITCH,  RET/NOP      RET = NO Trace, NOP = Trace
         .            (Editors note: be careful here, the
         .            label SWITCH has been used else-
         .            where in SCELBAL!)
         LLI 340      Point to line number buffer.
         CAL TEXTC    Display line number.
         LAI 001      Set up number of blanks.
         CAL TABC     Display blank.
         LLI 201      Replace SYNTX3 instructions.
         LBM
         RET          Return to SYNTX3.

        [AT PAGE 07 LOCATION 074 SET UP:]
         JTZ UDF(*)   Jump to UDF function.

        [AT A SUITABLE PATCH AREA ADD:]
UDF(*),  LLI 126      Point to MSB of FPACC.
         LHI 001
         LAM          Get MSB.
         CPI 100      Compare for a FPFIX "1."
         LLI ***      Address of SWITCH point
         LHI ***      For TRACE switch.
         JTZ TRAC     If comparison = 0 move a NOP
         LMI 007      To the switch, else move a RET
         RET          to the switch. Then exit.
TRAC,    LMI 300      Set up a NOP for the switch.
         RET          Exit.
```

ONE MORE TIME

In SCELBAL UPDATE Issue 04 of 1/77 on page 03 Mr. James Tucker of 3 Grove Street, Exeter, NH 03833 discussed a problem with storage of the first variable in the variables symbol table. He recently wrote to notify us of a related problem and a proposed correction:

"The program as it now functions skips the first storage cell when the first variable encountered is a "FOR—NEXT" variable. The search for this variable counts through the variables symbol table and gets part way through the page *(on which the variables are stored — Ed.)* again before finally finding the variable it seeks in the FOR—NEXT stack."

Mr. Tucker submitted two patches shown here "that look for an empty variables symbol table. If empty, a jump avoids advancing the pointer."

```
Present program:
010 132   106 356 022        CAL SWITCH

Change to:
010 132   104 052 075   **   JMP PATCH  (or suitable loc)

And put in the following patch:
075 052   106 356 022        CAL SWITCH
075 055   307                LAM
075 056   074 000            CPI 000
075 060   110 135 010        JFZ 010 135  (return)
075 063   106 356 022        CAL SWITCH
075 066   104 201 010        JMP STOSY3A


Present program:
005 065   106 356 022        CAL SWITCH

Change to:
005 065   104 017 075        JMP PATCH  (or suitable loc)

And put in the following patch:
075 017   106 356 022        CAL SWITCH
075 022   307                LAM
075 023   074 000            CPI 000
075 025   110 070 005        JFZ 005 070  (return)
075 030   106 356 022        CAL SWITCH
075 033   104 134 005        JMP LOOKU2A
```

SCELBI COMPUTER CONSULTING INC.
Post Office Box 133 PP STN
Milford, CT 06460

U. S. POSTAGE
FIRST CLASS
PAID
PERMIT No. 481
MILFORD, CONN.

FIRST CLASS MAIL!

SHIP TO

AME: _____

DDR: _____

ITY: _____

ZIP

Please note: All foreign orders must be accompanied by bank or postal money order payable in U.S. funds.

INDIVIDUALS SHOULD SUBMIT PAYMENT IN FULL BY CHECK OR MONEY ORDER — OR USE YOUR "MASTERCHARGE" ACCT.
"MASTERCHARGE"
ACCOUNT NUMBER .....

BANK # ..... EXPIRATION DATE .....

SIGNATURE .....

QTY	DESCRIPTION	UNIT PRICE		TOTAL PRICE	

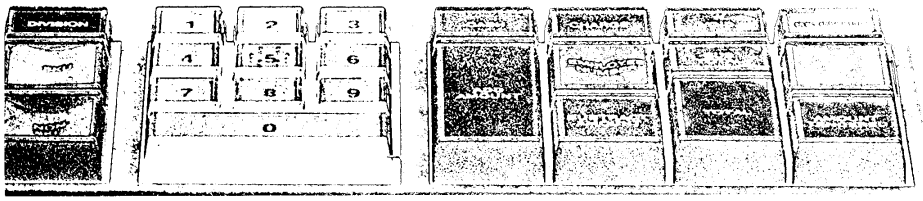
MERCHANDISE TOTAL .....

CONNECTICUT RESIDENTS ADD 7% STATE TAX .....

SHIPPING CHARGES .....

AMOUNT ENCLOSED .....

ORDER FORM



INTRODUCING SCELBI's 8080 STANDARD EDITOR

Here, at last, is an efficient way to edit text when preparing program source listings or other text material. You'll need an 8080 computer, with a minimum of 2K memory (of which at least 1K should be RAM); a text input device, like a keyboard; and a display/text output device.

OPTIONAL HARDWARE

Additional memory beyond 2K allows expanded text buffer storage area. Recommend 4K-8K for practical applications. Bulk storage I/O devices allow text to be saved for future use/modification.

SOFTWARE REQUIRED

User provided I/O driver routines for whatever I/O devices will be utilized. Each I/O device is linked to the program by a single vector for ease in adapting the program to individual systems.

MEMORY UTILIZED

The assembled listing provided in the manual resides in pages 01 through 05 (hexadecimal which is 001 through 005 octal). Pages 00, part of 05, and all of 06 (hexadecimal-000, part of 005 and all of 006 octal) are left available for user provided I/O routines. Pages 07 (hexadecimal-007 octal) through available memory used for text buffer.

OPTIONAL PAPER TAPE NOW AVAILABLE

An optional object code on punched paper tape is available. Specify 8080ED-OPT, $6.00. And you can order optional commented source listing on paper tape too. Specify 8080ED-SPT, $20.00.

MNEMONICS UTILIZED

This program is written in 8080 machine language standard industry accepted mnemonics for the 8080 CPU (such as MOV A, B; INX H; CALL; etc.) (Note: SCELBI is discontinuing its use of special 8080 compatible mnemonics which have characterized its 8080 programs in the past.)

PROGRAM OPERATION

This is a standard line-oriented text editing program intended for use in the creation of source listings and similar text manipulations. The program operates in two modes; the Text Entry mode for entering text into the text buffer and the Command mode used to specify operator directives. Information in the text buffer may be manipulated using the Command directives and the contents of the text buffer transferred to an external storage device or filled from an external storage device.

PROGRAM COMMANDS

APPEND (A) text to the text buffer; CHANGE (C) text; DELETE (D) text; INSERT (I) text; LIST (L) text; character SEARCH (S); READ (R) from or WRITE (W) to an external storage device; CLEAR text buffer; plus single character deletion, tab (spacing), and various character search directives.

DOCUMENTATION

In the famous SCELBI tradition. The program manual describes the operation of the editor, presents detailed discussions of all major routines with flow charts, contains two completely assembled listings (one with addresses and object code in hexadecimal notation and one in octal notation), and of course includes operating instructions and tips on enhancing the program if desired.

SPECIAL FEATURES

Because the program has been carefully organized and written with all memory references assigned labels, it may be readily reassembled to reside in any general area in memory. This program may even be assembled to reside in just 1K of ROM provided that some RAM area is available for scratch pad and text buffer use!

OPTIONS

A punched paper tape of the object code for this editor (as described in the documentation) is available. The object code tape is provided in the widely accepted "hexadecimal format." Also, the complete, commented source listing of the program as presented in the documentation is available in straight ASCII format on punched paper tape. Fan-fold paper tapes are provided for ease in handling. Additionally, opaque paper tape is supplied to facilitate the use of low cost optical paper tape readers now in widespread use. NOTE: Paper tapes are sold only as optional supplements to the documentation.

ORDER YOUR COPY TODAY...$12.95

SCELBI COMPUTER CONSULTING INC.

Post Office Box 133 PP STN
Milford, CT 06460

Price shown for North American customers. Master Charge, Postal and Bank Money Orders preferred. Personal checks delay shipping up to 4 weeks. Pricing, specifications, availability subject to change without notice. SCELBI Books are available in many fine Computer Stores. **IMPORTANT!** Include 75¢ postage/handling for each item delivered by U.S. Mail Book Rate, or $2 for each item shipped First Class or via UPS.

CONTENTS

INTRODUCTION

SECTION ONE

SECTION TWO

SECTION THREE

SECTION FOUR

SCELBI COMPUTER CONSULTING INC.
Post Office Box 133 PP STN
Milford, CT 06460

Dear Small Computer User:

We are enclosing information about two of our newest SCELBI software products — the SCELBI 8080 STANDARD ASSEMBLER and the SCELBI 8080 STANDARD EDITOR. These two programs are essentially re-writes and enhancements of our previous 8080 Assembler and Editor programs — the primary difference being that these programs are written in, and the Assembler Program processes, industry standard mnemonics. [The mnemonics originally promulgated by Intel Corporation. developer of the 8080 CPU and now widely accepted in industry.] Previous SCELBI Assembler and Editor programs utilized a set of mnemonics that were an extension of the commonly used 8008 instruction set. Since the 8008 is no longer in common use, SCELBI is discontinuing its use of the special mnemonics that has characterized its 8080 programs in the past.

Many people currently inquiring about SCELBI products are not aware of our long history in the microcomputer field. SCELBI is a Connecticut Corporation formed in 1973 devoted to producing quality microcomputer products. Since then we have gain world-wide recognition for our extremely well documented software. Users tell us it is the best documented software one can buy at any price. Even better for you is the fact that since thousands of people buy it, the price is often lower than our competitor's undocumented software!

What is so great about documented software? If you are into computers, we don't have to tell you. Proper documentation saves you time in implementing the software — and it gives you the freedom to modify or alter the software to meet your specific needs — a luxury that is not even practically possible in undocumented programs of this nature.

How thorough is our documentation? To give you an idea, take a look at the tables of contents for our SCELBI 8080 STANDARD ASSEMBLER and SCELBI 8080 STANDARD EDITOR which appears on the reverse side of the accompanying product description literature. From it, you can not only observe that we provide completely assembled and *fully commented* source listings (and in two versions — one with hexadecimal addresses and object code, the other with octal notation!), but, you can also see that we provide additional descriptiive and instructional information on each major portion of the program; information on where and how to insert your I/O routines; how to implement and use the program; and even suggestions for expanding its capability. SCELBI software is educational as well as practical. Read our program manuals and you will know exactly how and why a program such as an Assembler or Editor operates. Our program manuals are laid out for efficiency too! From the precise descriptions of routines, to critical flowcharts, to alphabeticized symbol table listings, to assembled commented source listings, to operating instructions — they are designed to enable the user to rapidly learn what one needs to know to successfully implement the program — customized to the individual user's requirements.

If your already a SCELBI customer, then you know the worth of our products, and we hope the technical information contained in the enclosed literature convinces you to add to your SCELBI library. If you are not a previous SCELBI customer — you have been missing a lot. Take a close look at the enclosed literature. We know how to deliver software: with documentation!

Yours sincerely,

Richard Hess

P.S.: There is a convenient to use order form on the back that will allow us to process your order quickly and efficiently. We have also enclosed a pre-addressed envelope for your convenience. Have a nice day!

..... YES! I would like a copy of the program manual the SCELBI 8080 STANDARD ASSEMBLER at the price of $19.95 plus 75¢ shipping/handling by U.S. Mail for a total of $20.70.

..... While your at at, send me a 8080SA–OPT object code paper tape at $10.00 plus 75¢ shipping/handling by U.S. Mail for a total of $10.75.

..... I would also like to have the 8080SA–SPT source listing paper tape at $39.00 plus 75¢ shipping/handling by U.S. Mail for a total of $39.75.

..... Your SCELBI 8080 STANDARD EDITOR program manual is what I need. Please send me a copy at $12.95 plus 75¢ shipping/handling by U.S. mail for a total of $13.70.

..... I would also like the 8080SE–OPT object code paper tape at $6.00 plus 75¢ shipping/handling by U.S. Mail for a total of $6.75.

..... And, I would like the 8080SE–SPT source code paper tape at $20.00 plus 75¢ shipping/handling by U.S. Mail for a total of $20.75.

PLEASE NOTE THAT ABOVE PRICES ARE FOR U.S./CANADIAN DELIVERY BY MAIL ONLY!
(PLEASE ADD 1.25 *PER ITEM* FOR UNITED PARCEL SERVICE OR SPECIAL DELIVERY SERVICE.)

..... I enclose a total of ..... by: .... Check .... Money Order

..... I want to charge this order to my credit card: .... VISA .... MASTERCHARGE. Card info below:

CARD ACCOUNT NUMBER:.....

Card Holder's Signature:.....

If applicable, please give the card Bank Number:....., and card Expiration Date:.....

NAME:.....

STREET:.....

CITY:..... STATE:..... ZIP:.....

IMPORTANT

We regret that time must be allowed for personal checks to be processed and cleared before shipment can be made. For speediest service we recommend the use of Bank or Postal Money Orders or a VISA or MASTERCHARGE credit card.

SCELBAL — II

SCELBAL-II is a revised version of SCELBI'S original edition of SCELBAL specifically written to run on 8080 (or Z—80) systems. As previous users of SCELBAL know, the language was initially developed to execute on 8008 based systems. It was thus compatible with, but not efficient when running on, 8080 based systems. An attraction of the the language when originally introduced, besides its outstanding documentation, was that people acquiring it to run initially on a 8008 based unit would be able to transfer it to an 8080 system while maintaining complete high level program compatability.

Since the time SCELBAL was created, the once ubiquitous 8008 CPU has become all but extinct. Since SCELBAL was originally developed for the 8008 CPU, it was written in the mnemonic language originally promulgated for that device. The 8080 version of SCELBAL was an exact copy of the 8008 one at the assembly language level. Only the machine code (object code) was changed to conform to the requirements of the 8080 CPU. Because of various 8008 limitations, the 8080 version of SCELBAL was relatively inefficient in operation.

SCELBAL-II was written to eliminate much of the inefficiency when SCELBAL is run on an 8080 or 8080 compatible machine (such as a Z—80). SCELBAL-II is written in industry standard 8080 mnemonics (as promulgated by the original developer of the 8080 CPU — Intel Corporation) which can be assembled by our new SCELBI 8080 STANDARD ASSEMBLER or other similar assemblers available on the market.

Of course, SCELBAL-II takes advantage of many of the extended instruction set capabilities of the 8080 that were not available on the 8008 CPU. This has allowed the interpreter to be improved in several areas of its operation while considerably reducing the amount of memory required by the interpreter. In fact, several additional capabilities have been added to the fundamental package which were not provided in the original version of SCELBAL.

Specifically, SCELBAL-II has the following capabilities that were not provided in the original version of the language: Three new statements — DATA, READ and RESTORE which allows the user to specify data as part of a program and access this data under high level program control using the READ and RESTORE directives. Essentially unlimited regular variables (actually limited by the amount of memory available in a system and the maximum number of permissable two character label "names") instead of the limit of 20 provided in the original version of the language. And, Improved LISTing capability that allows a specific line or series of lines to be listed from the User's Program Buffer (instead of having to list the entire buffer). Furthermore, these increases in capability were achieved while providing a typical overall operating speed improvement of three-fold and reducing memory requirements substantially. Now, the entire fundamental package resides in 6.5K of memory — and we stuck to our policy of not using page zero! Furthermore, temporary storage locations formerly split over two sections of memory have been regrouped to occupy three consecutive pages at the top of the operating routines (or just below the user buffer area). This, coupled with the fact that symbolic labeling has been used extensively, makes re-assembling the program to reside in other areas of memory (other than the one provided in our assembled listing — pages 01 through 1A hexadecimal, which is 001 through 032 octal) a relatively straightforward project!

SCELBAL-II does NOT include extended functions, and we have no current plans to provide those capabilities. Naturally, those users who have added those capabilities to the original version of SCELBAL may wish to adapt such capabilities to SCELBAL-II on their own.

SCELBAL-II will be sold only to purchasers of SCELBAL and will be released as an assembled, uncommented source listing in hexadecimal format (addresses and object code in the assembled

(O V E R)

listing use hexadecimal notation). Mnemonics utilized are industry standard 8080 mnemonics such as MVI, MOV, etc., as SCELBI has discontinued its use of its earlier blend of 8008/8080 mnemonic names. An optional punched paper tape (hexadecimal format) of the object code will be available to purchasers of the listing. Additionally, the source code will also be optionally available on punched paper tape.

The general organization and structure of the program remains similar to that of the original version of SCELBAL and the original SCELBAL publication will be considered the fundamental explanatory text for the program. There have been some changes in key routines to increase the operating speed and efficiency of the interpreter, but experienced machine language programmers familiar with the workings of SCELBAL should be able to discern the key differences on their own. Highlights of the major alterations will be pointed out in the SCELBAL-II manual containing the listing.

We wish to make it perfectly clear that SCELBAL-II will be essentially an unsupported supplement to SCELBAL. Purchaser's will be on their own. There will be no SCELBAL UPDATES or similar continued support of this version as was the case with SCELBAL.

The price of the uncommented source listing of SCELBAL-II with addresses/object code in hexadecimal format is $20.00 to registered SCELBAL owners. YOU MUST GIVE YOUR REGISTRATION NUMBER WHEN ORDERING (found on the cardboard insert as the last page in the book) in order to receive this price. ALL ORDERS ARE SUBJECT TO OUR ACCEPTANCE AND VERIFICATION OF REGISTRATION. The price of the listing of SCELBAL-II to others is $69.00 — for which they will also receive a copy of the original SCELBAL publication!

A paper tape of the object code for SCELBAL-II in hexadecimal format on fan-fold black tape (works wonders for those low cost optical tape readers) is available as an option to those purchasing the listing for $15.00. For those ambitious users who really want to get into things, a paper tape of the uncommented source listing (ASCII source format) will be available to those purchasing the listing for an additional $35.00. (The listing is provided in the form of a brief manual which includes other goodies such as a symbol table listing and other useful information not included on the source paper tape.)

..... YES! Please send me the uncommented source listing manual for SCELBAL-II.

.... My SCELBAL Registration Number is ....., I get the SCELBAL-II listing for $20.00!

.... I am not a previous SCELBAL owner, so send me SCELBAL and SCELBAL-II for $69.00.

..... Also send me the object code for SCELBAL-II on punched paper tape (SCELBAL-II–OPT) for $15.00.*

..... I'd also like the source listing for SCELBAL-II on punched paper tape (SCELBAL-II–SPT) for $35.00.*

* Paper Tapes sold only in U.S. and Canada.

NOTE: PLEASE ADD 75¢ for each item shipping/handling in U.S. and Canada by Mail
And, Connecticut residents please include 7% State Sales Tax

I enclose a total of:..... by: .... Check .... M.O. .... Charge to VISA .... Charge to MASTERCHARGE

CARD ACCOUNT NUMBER:.....

Card Holder's Signature:.....

If applicable, please give the card Bank Number:....., and card Expiration Date:.....

NAME:.....

STREET:.....

CITY:....., STATE:..... ZIP:.....

CONTENTS

INTRODUCTION

SECTION ONE

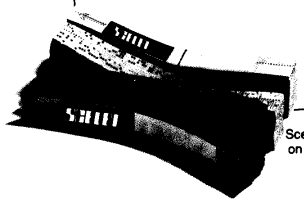
SECTION TWO

SECTION THREE

SECTION FOUR

SCELBI's new '8080' STANDARD ASSEMBLER

FUNCTION:	Assembles programs written in symbolic language for an 8080 CPU on an 8080 based system.
HARDWARE REQUIRED:	8080 computer with minimum of 4K memory (of which at least 1K should be RAM); a source listing input device; an object code output device.
OPTIONAL HARDWARE:	A system console device such as a keyboard/CRT or keyboard/printer will allow convenient control of the program using executive commands; additional memory beyond 4K will allow expanded symbol table length, or capability to assemble directly into memory.
SOFTWARE REQUIRED:	User provided I/O driver routines for whatever I/O devices will be utilized. Each I/O device is linked to the program by a *single* vector for ease in adapting the program to individual systems.
MEMORY UTILIZED:	The assembled listing provided in the manual resides in pages 01 through 0A (hexadecimal – 001 through 012 octal). Pages 00, part of 0A, all of 0B and 0C (hexadecimal – 000, part of 012, 013 and 014 octal) are left available for user provided I/O routines. Pages 0D (hexadecimal – 015 octal) on up used for symbol table storage (or as direct assembly areas in systems with sufficient memory).
MNEMONICS UTILIZED:	This program is written in, and accepts for assembly purposes, standard industry accepted mnemonics for the 8080 CPU (such as MOV A,B; INX H: CALL; etc.) [Note: SCELBI is discontinuing its use of special 8008 compatible mnemonics which have characterized its 8080 programs in the past.]
PSEUDO-OPERATORS:	Accepts the ORG (originate), END (stop assembly), SET (define a name), DB (data byte), DS (data string) and DW (data word or double byte) pseudo-operators.
PROGRAM OPERATION:	The program processes a source listing in two passes to produce assembled object code. An optional third pass allows an assembled listing to be obtained. Listings may be obtained in hexadecimal or octal format. The program will also display the contents of the symbol table at the operators request. The program can process source listings as single or multiple files. Program operation may be controlled from a console device using executive commands or through computer panel switches by jumping to appropriate locations within the program.
SOURCE FORMAT:	Convenient, easy to use, variable length fields permitted. Labels may be 1 to 6 characters in length, accepts both hexadecimal and octal numbers with or without leading zeros, has "literal" capability (can accept ASCII characters directly as data), allows use of letters of numbers as CPU register operands.
DOCUMENTATION:	Thorough – in the SCELBI tradition! The program manual describes the operation of the assembler, presents detailed discussions of all major routines, and contains two completely assembled listings (one provided in hexadecimal and one in octal notation). Of course it includes operating instructions and even provides a routine that may be used for loading programs produced by the assembler!
SPECIAL FEATURES:	Because the program has been carefully organized and written with all memory references assigned labels, it may be readily reassembled to reside in any general area in memory. It may even be reassembled to reside in ROM provided that some RAM area is available for scratch pad and symbol table use!
OPTIONS:	A punched paper tape of the object code for this assembler (as described in the documentation) is available. The object code tape is provided in the widely accepted "hexadecimal format." Also, the complete, commented source listing of the program as presented in the documentation is available in straight ASCII format on punched paper tape. Fan-fold paper tapes are provided for ease in handling. Additionally, opaque paper tape is supplied to facilitate the use of low cost optical paper tape readers now in widespread use. NOTE: Paper tapes are sold only as optional supplements to the documentation.



Scelbi's 8080 Standard Assembler: **$19.95** Optional object code on punched paper tape, specify 8080SA-OPT: **$10.00.** Optional commented source listing on punched paper tape, specify 8080SA-SPT: **$39.00.**

OPTIONAL PAPER TAPE NOW AVAILABLE!

SCELBI COMPUTER CONSULTING INC.
Post Office Box 133 PP STN
Milford, CT 06406

revised 2nd Edition...with industry standard ...has just arrived! Plus, a special bonus!

Scelbi "8080" Software Gourmet Guide & Cook Book

Scelbi has done it again. Here is the great book that lets you cook up appetizing, hot programs for your own '8080'. And, now, it's been completely revised, and republished, to reflect the latest industry standard mnemonics.

This tasty delight of all new second generation software for your '8080' is chock full of mouth-watering applications to satisfy the most fussy palate. It's a mouthwatering delight.

Over 200 pages. Chock full of action, facts.

You'll devour the contents of this action-packed, information-stuffed book. Gobble up practical 'how to' facts. The description of the '8080' instruction set. How to manipulate the '8080' stack. Flow charts and source listings. General purpose routines for multiple precision operation. Programming time delays for real time applications. Random number generators. And many, many more basic programming functions you can sink your teeth into.

Three versions now available.

That's right. Now there are three versions of this great book. The DeLuxe Hardcover edition, a must for any library, is only $14.95. The new heavy-duty Vinyl covered edition, for the hard working, heavy computer user, is only $12.95. The same facts are packed into an economical paperback edition for a low, low $10.95.

Floating Point Arithmetic Routines.

The new 2nd edition of the famous Scelbi *Cook Book* even includes completely assembled floating point arithmetic programs (worth the price of the book alone!) . . . plus input/output processing for your basic I/O programming through interrupt processing. There are code and numeric conversion routines. Real time programming. Search and sort routines. And too many finger-lickin' goodies to mention here.

'8085' users will find a fountain of champagne facts in here too.

'8080' Pocket Guide Bonus offer.

If you act right now you can own Scelbi's equally famous *'8080' Programmer's Pocket Guide*, a compact, pocket-size, instant reference that explains the instruction set in detail (duplicates Chapter 1 of the *Cook Book*), giving both octal and hexadecimal codes. It normally sells for $2.95. Order your copy of *Scelbi "8080" Software Gourmet Guide & Cook Book* now, and the *Pocket Guide* is yours for only $2.00!

Don't delay. Order your copy today.

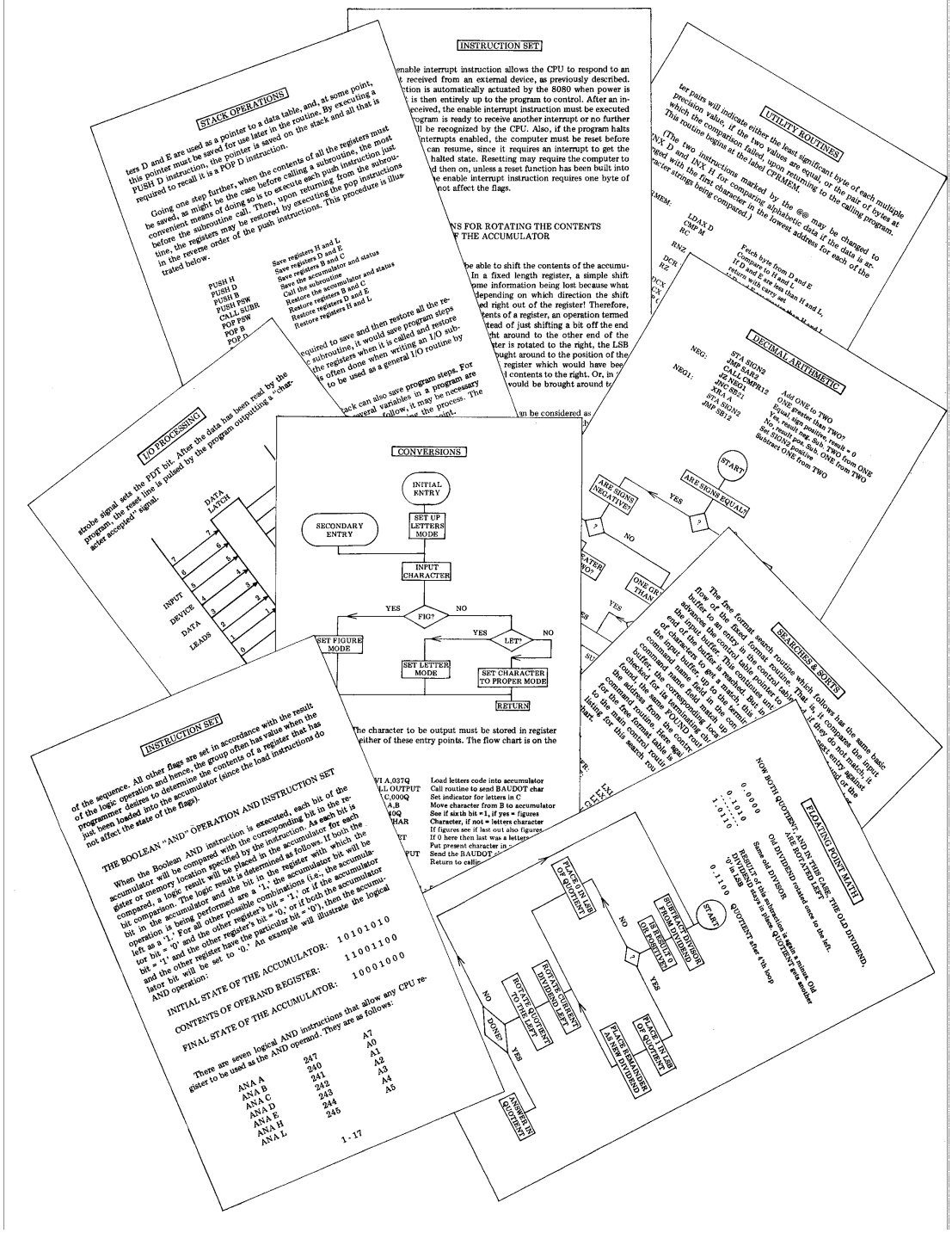
You've heard about this great book. Here's your chance to own the updated, current edition in one of three versions to fit your budget. It's everything the computer chef needs. And, you can't beat the price . . . or the bonus offer. Order your copy today. *Scelbi's "8080" Software Gourmet Guide & Cook Book.* Write to Scelbi Computer Consulting, Inc.; P.O. Box 133 PP STN, Milford, CT 06460.

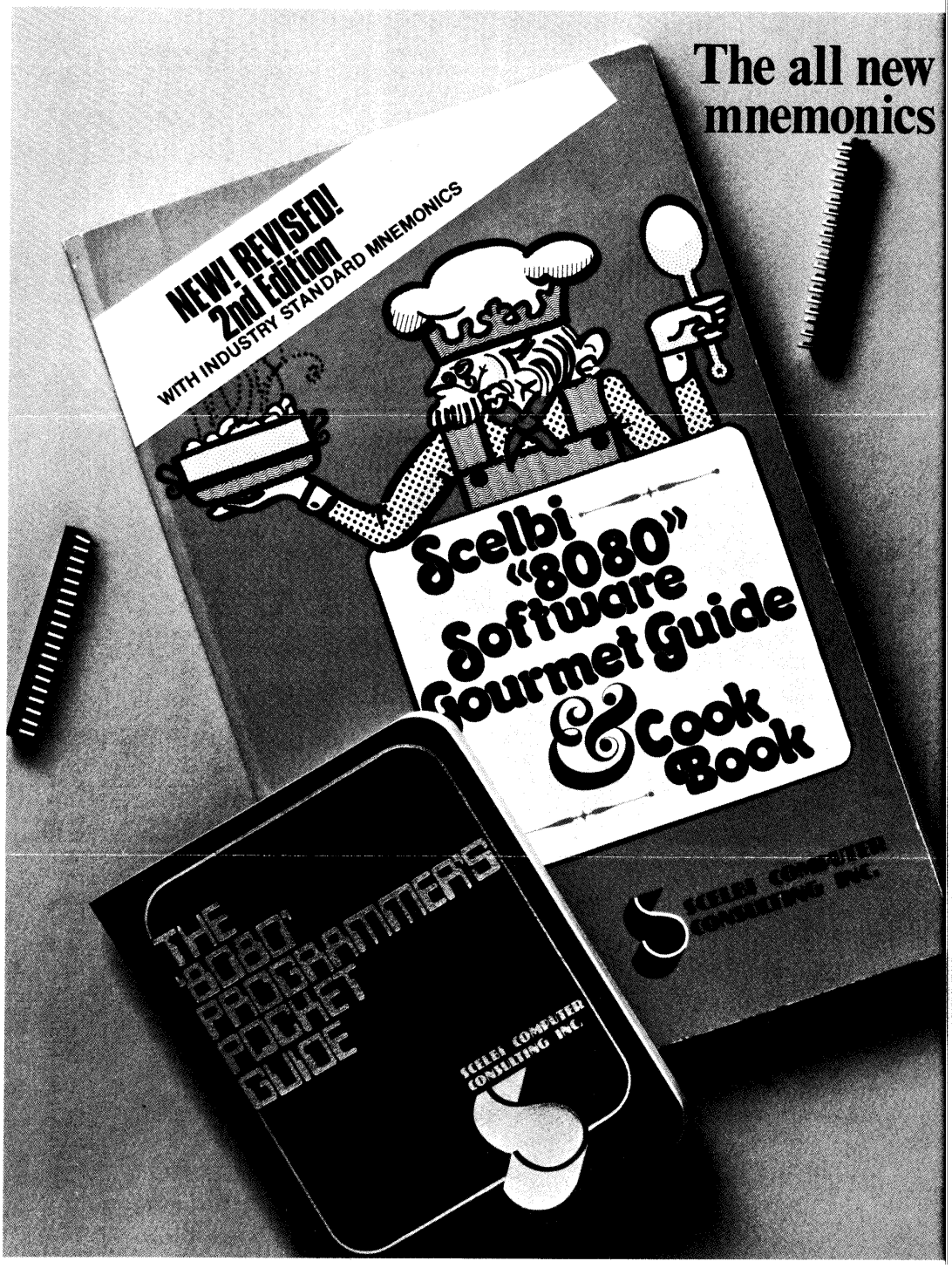
SPECIAL BONUS OFFER!

Act now and you can own Scelbi's popular *'8080' Programmer's Pocket Guide*. This compact book, 3"x4½", slips into your pocket, so you can carry it with you everywhere. It's an ever-ready instant reference for operation of your '8080'. It explains the instruction set in painstaking detail, and gives both octal and hexadecimal codes. This valuable book regularly sells everywhere for $2.95. If you order your 2nd Edition *Cook Book* now, you can also purchase the *Pocket Guide* for a low, low $2.00. Do it now!

Price shown for North American customers. Master Charge, Postal and Bank Money Orders preferred. Personal checks delay shipping up to 4 weeks. Pricing, specifications, availability subject to change without notice. SCELBI Books are available in many fine Computer Stores. **IMPORTANT!** Include 75¢ postage/handling for each item delivered by U.S. Mail Book Rate; or $2 for each item shipped First Class or via UPS.

- **DeLuxe Hardcover Edition, now only $14.95**
- **Durable Vinyl covered Edition, now $12.95**
- **Economical Paperback Edition, now $10.95**


STACK OPERATIONS
INSTRUCTION SET
UTILITY ROUTINES
I/O PROCESSING
DECIMAL ARITHMETIC
CONVERSIONS
SEARCHES & SORTS
INSTRUCTION SET
FLOATING POINT MATH
INITIAL ENTRY
SECONDARY ENTRY
SET UP LETTERS MODE
INPUT CHARACTER
FIG?
LET?
SET FIGURE MODE
SET LETTER MODE
SET CHARACTER TO PROPER MODE
RETURN
START
ARE SIGNS NEGATIVE
ARE SIGNS EQUAL
DONE?
ANSWER IN QUOTIENT
PLACE 1 IN LSB OF QUOTIENT
PLACE 0 IN LSB OF QUOTIENT
ROTATE QUOTIENT TO THE LEFT
ROTATE CURRENT DIVIDEND LEFT
SUBTRACT DIVISOR FROM DIVIDEND
IS RESULT 0 OR POSITIVE?
LEAVE REMAINDER AS NEW DIVIDEND
DATA LATCH
INPUT DEVICE DATA LEADS


The all new mnemonics
NEW! REVISED! 2nd Edition
WITH INDUSTRY STANDARD MNEMONICS
Scelbi "8080" Software Gourmet Guide & Cook Book
SCELBI COMPUTER CONSULTING INC.
THE 8080 PROGRAMMERS POCKET GUIDE
SCELBI COMPUTER CONSULTING INC.